Leveraging Stochasticity for In-Situ Learning in Binarized Deep Neural Networks

Steven D. Pyle, Justin D. Sapp, and **Ronald F. DeMara,** University of Central Florida

A recent thrust in Deep Neural Network (DNN) research has been towards binary approaches for compact and energy-sparing neuromorphic architectures utilizing emerging devices. However, approaches to deal with device process variations and realization of stochastic behavior intrinsically within neural circuits have remained underexplored. Herein, we leverage a novel probabilistic spintronic device for low-energy recognition operations which improves DNN performance through active in-situ learning via mitigation of device reliability challenges.

Deep Neural Networks (DNNs) have realized impressive feats of intelligence, surpassing humans in specific tasks and achieving high efficacy at speech recognition, machine translation, and higher-level human-like reasoning activities such as interpretation and classification of visual art [1]. Although there are many successful architectural models used by DNNs, their common characteristic is the use of many layers of hidden nodes to realize "deep" topologies of non-linear neurons with linear weighted connections trained by backpropagation to distinguish complex inputs with high degrees of accuracy [2]. However, this beneficial characteristic of having many layers results in high computational demands and a large memory footprint [3]. Thus, recent works into reducing the computation and memory overheads of DNNs have investigated the possibility of attaining similar recognition capabilities using reduced-precision approaches which incur significantly lower computation and memory demands. By reducing these overheads, in-situ networks could potentially be realized on resource-constrained platforms such as mobile and Internet of Things (IoT) devices [4].

Promising advancements towards this goal have focused on the substitution of high-precision floating-point parameters with binary representations, which replace expensive multiply-and-accumulate computations with bitwise logical operations and bit counting. These Binary Neural Networks (BNNs) have replicated some incredible feats of narrow intelligence demonstrated by Deep Learning with energy profiles that could be implemented on more resource-constrained systems by using efficient custom neuromorphic hardware with emerging computing devices [5-9]. This has led to the development of neuromorphic hardware accelerators that can implement such networks in a highly efficient manner [5-8]. The work presented herein extends these works by utilizing an emerging compact stochastic device, called the probabilistic bit (p-bit) [10], that naturally implements a non-linear Probabilistic Activation Function (PAF). The low-current operation of the p-bit allows for a seamless integration with low-voltage and high-resistance Resistive Random Access Memory (RRAM) crossbar and pseudo-crossbar arrays, which leads to very low energy consumption per computation. In addition to the novel PAF proposed herein, we analyze how process variations in RRAM devices impact the performance of BNNs that are implemented using our scheme, and how such variations can be mitigated with in-situ training.

To summarize, this work provides the following contributions:

1) We demonstrate the feasibility of a compact PAF that uses just 4.98 μW combined with parallel binary RRAM pseudo-crossbar arrays for a low power of 75 nW per each weighted connection having an excitatory input, and

2) We evaluate the effects of RRAM process variation rates up to 50% on the recognition rate for the CIFAR-10 image recognition dataset using a convolutional neural network and demonstrate how in-situ learning can mitigate the resulting performance degradation.

DEEP NEURAL NETWORK ACCELERATION

Convolutional Neural Networks (CNNs), the popular class of DNNs that we investigate herein, are multi-layered networks that typically consist of several convolution layers, which convert high dimensional data, such as RGB images, into features, followed by a number of fully connected layers and terminated with a Log SoftMax layer for classifying the input data objects into labels based on their features, as depicted in Figure 1 [2]. Within each layer, either convolutional or fully-connected, the primary computations are realized by abstract neurons that calculate the weighted-summation of their input connections. Each neuron then computes a non-linear activation function of that weighted-sum. There are many different activation functions used with DNNs, such as *rectified linear units*, *tanh*, *sigmoid*, and others. Since DNNs traditionally utilize high precision floating-point representations of millions and sometimes billions of parameters, they incur significant memory requirements and computational operations during their training and deployment phases. They also are subject to the high latency overhead of transferring data between the memory and processor in traditional von-Neumann architectures. Thus, the development of BNNs which discretize the weights and activations of DNNs to binary values, as shown in Figure 1, can greatly reduce the memory and computation

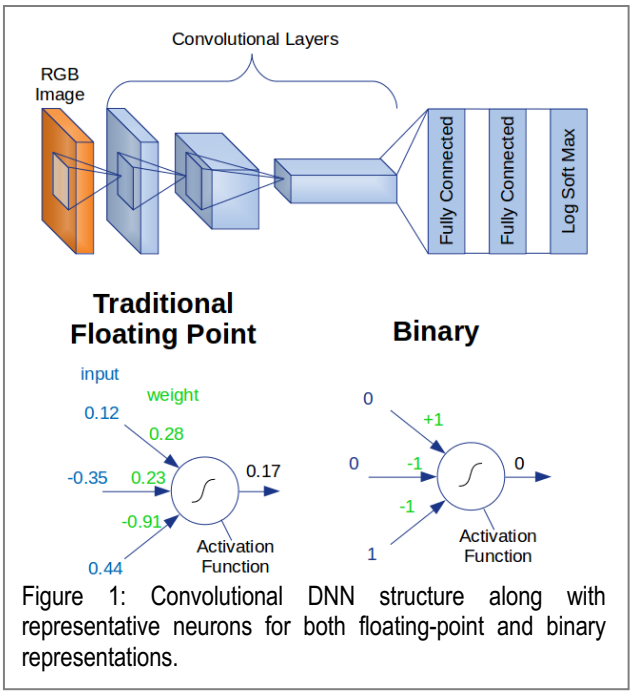

Figure 1: Convolutional DNN structure along with representative neurons for both floating-point and binary representations.

overheads of training and utilizing DNNs. Namely, the expensive floating-point operations can be simplified into highly-efficient bitwise computations using bit-counting [5]. If we realize these memory and computation overhead reductions by implementing BNNs with custom neuromorphic accelerators using resistive devices in crossbar and pseudo-crossbar topologies [6-8], we can also reduce physical chip area, energy, and computation time requirements, which we expound upon in the following sections.

The binarization of weights in BNNs are typically constrained to {-1, +1} values, while activation functions have been explored with both {0, 1} and {-1, +1} constraints [6-8]. Additionally, the activation function is typically implemented with a deterministic sign function due to its straightforward implementation on most hardware [6-8]. Stochastic binarization has been proposed as more appealing than deterministic binarization approaches [5], although very few works have investigated PAFs in BNNs. This is primarily due to the inherent difficulty, device count, and wiring complexity in implementing PAFs using standard CMOS-only approaches. These can be beneficial for modeling and abstracting complex dynamic distributions, as identified broader literature [11]. Another consideration for BNN accelerators using resistive crossbar arrays is the effect that process variation has on the accuracy of the network. With deviations from the device's ideal resistance values, the weights effectively shift from their intended values, which as we show later, can cause a significant increase in the error rate. However, we demonstrate that by incorporating the hardware into the neuromorphic training loop, it is possible to mitigate almost all degradations of accuracy associated with process variation. This is detailed in Figure 2, where Figure 2a shows the propagation of activations (x) between layers, which is computed using the accelerator circuitry for in-situ training or ideal values in software using off-platform training, and Figure 2b shows the training loop for both in-situ and off-platform approaches where the error gradient (e) calculations and subsequent weight updates are performed in a connected CPU or ASIC. For the in-situ approach, the RRAM-based synaptic weights are physically updated, and these weights are used to compute activations in the next forward pass. For off-platform training, ideal software-based weight values are updated and used to calculate the activations based on ideal behavior.

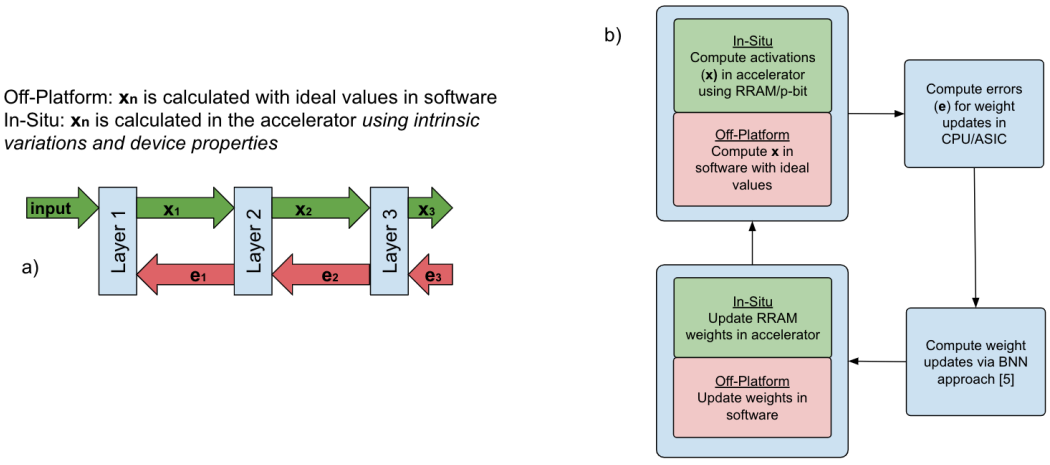

Figure 2: Training illustration for in-situ and off-platform methods. a) propagations of activations (**x**) and error gradients (**e**) used for calculating weight updates during training. b) the training loop with differences between in-situ and off-platform training highlighted.

RECENT WORK

Several recent works have aimed towards realizing BNN acceleration through the utilization of emerging devices, such as RRAM and spintronics, to compute the necessary binary operations *in-memory*. This frees up chip area and eliminates bottlenecks in the data pathways between memory and computational resources. A selection of these works in Table 1 is compared on the basis of Transistor Count, which determines the silicon die area that is needed to interface with and facilitate computational operations in the memory array, the Sequential/Parallel operation of the in-memory computations, and the Variation Degradation Factor. The latter quantifies the accuracy degradation, defined as the increase in mean percent recognition error across the CIFAR-10 dataset divided by the percentage value of single-sigma variation in resistance of the RRAM elements, as described subsequently herein. The influential work of Sun et al. proposed an RRAM-based XNOR BNN that is capable of both sequential operation (computing the XNOR of inputs and weights one input bit at a time, summing the

result, and then applying the sign activation function) and parallel operation (using two input lines per input bit and two single-transistor/single-resistive-element (1T1R) cells per weighted input to compute parallel bitwise XNOR operations and then using a Sense Amplifier output to realize the sign activation function) [6]. Ni et al. proposed a sneak-path-free binary crossbar using two types of RRAM devices that do not require a select transistor for each bit cell, and their activation function is determined by a voltage comparator, which uses 16 transistors [8]. They found that a 29% variation rate in the resistance values of the bit cells degraded accuracy by 4%, leading to a Variation Degradation Factor of 0.138. The work presented herein uses a compact, low-power PAF and an RRAM array to conduct parallel BNN computations that are resilient to RRAM variations. With a Variation Degradation Factor of 0.02 when used with in-situ training, concerns about process variation are practically eliminated. Although additional works investigated mitigating the effects of process variations in non-binary RRAM crossbar approaches [12], comparisons herein are restricted to those utilizing binary encodings. Additionally, while FPGA-based accelerators offer off-the-shelf programmability and achieve orders of magnitude higher throughput in digits per second than CPUs on recognition benchmarks such as the MNIST dataset, significant energy-efficiency and area optimizations are sought beyond reconfigurable fabrics by using neuromorphic architectures. When adopting an in-situ accelerator-based approach towards this goal, the throughput during the training phase becomes bandwidth-limited only by the datapath between CPU/ASIC, RAM, and the accelerator itself. Thus, proper speed-matching among these components is essential. During the inference phase, throughput of a parallel crossbar remains comparable to similar binarized RRAM-based neural network approaches, which significantly exceed CPU, GPU, FPGA, or CMOS-ASIC approaches in-practice [8]."

Table 1: Neuron attributes of recent BNN approaches.

BNN Approach	Transistor Count	Sequential/ Parallel	Variation Degradation Factor
Sun et al. [6]	14	Mixed	N/A
Ni et al. [8]	16	Parallel	0.138
Angizi et al. [9]	>10	Sequential	N/A
Work Herein	*4*	*Parallel*	*0.02*

Accelerator Design

The neuromorphic accelerator described herein is depicted in Figure 3. It consists of multiple BNN layers, where each layer contains a pseudo-crossbar array, along with the associated PAFs at the outputs, and corresponds to either a convolution kernel or a fully-connected layer, as determined by the DNN architecture that is being implemented. In addition to the BNN layers, our simulations assume that a rudimentary on-board CPU or ASIC with access to sufficient RAM is used for handling the training logic and backpropagation calculations, as well as storing and delivering the training/test data and labels. Those resources do not significantly impact the computational burden, as the majority of the calculation workload is engaged when computing each DNN layer's weighted sums and activation functions.

Pseudo-Crossbar Operation

Each layer in the neuromorphic accelerator performs three computations in parallel between all input and output bits. The first computation is the bitwise multiplication of the binary input and the stored binary weight between each input and each output. This computation is realized using two complementary memristors per input/output pair as well as two signal lines per input, which is similar to resistive pseudo-

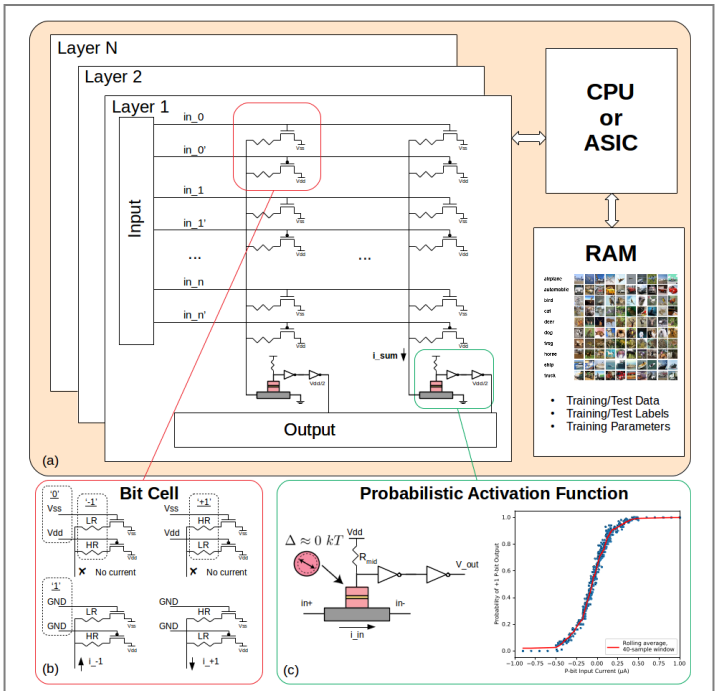

Figure 3: (a) Accelerated BNN-based classification of training and evaluation images stored within memory, using (b) weighted input values derived from a resistive memory/logic array, and (c) activation function based on a tunable, stochastic Spin Hall Effect MTJ with circular nanomagnet.

crossbars previously developed for processing BNNs [6-8]. As shown in Figure 3b, the paired input wire {*in_i, in_i'*} voltages signify a value of either '0' or '1'. The signal level '0' is represented by an input pair value of [*Vss, Vdd*] (here *Vss* is chosen to be *-Vdd*) which deactivates both the PMOS and NMOS transistors in the bit cell, allowing no current to flow regardless of the weight value. An input value of '1' is represented by an input pair value of [*GND, GND*], which activates both transistors, allowing current to flow through both branches. Each memristor pair corresponds to a '-1' or '+1' weight, depending on which memristor is in the High Resistance (HR) state and which memristor is in the Low Resistance (LR) state, as depicted in Figure 3b. When the input value is '1', meaning both transistors in the bit cell are on, the branch with the LR memristor sinks/sources the vast majority of the current flow in the bit cell, due to the large resistance ratio between the LR and HR states. Thus, if the LR branch connects to the NMOS and *Vss*, it will sink current, corresponding to an output value of '-1', and if the LR branch connects to the PMOS and *Vdd*, it will source current, corresponding to an output value of '+1'. Thus, the three possible output values of the bitwise input and weight multiplications are 0, -1, and +1.

The second parallel calculation performed in the BNN layer is the bit-counting operation, which is the summation of the parallel bitwise multiplications described previously along a single output path. This is accomplished by connecting the *in-* terminal of the p-bit to *GND* and the *in+* terminal to the bit line that connects all of the bit-cells within a single column, whereby the accumulation of currents according to Kirchoff's current law corresponds to parallel input-weight multiplications, which equates to one of the {0, -1, +1} current-level values described previously. The final calculation in the BNN layer uses the accumulated current summation from the previously described computation as the input to a PAF, which outputs a '0' or '1' with a probability according to a sigmoidal function as shown in Figure 3c. The implementation of the PAF is described next.

Concerning RRAM variability, in order to continually measure resistance over a period of time, there can be application of a read voltage to the memory cell and retention results for a 1 Kb array to maintain at least 5x resistance ratio at high temperature are published [16]. CBRAM readily provides R_on and R_off around 12Kohm and 100Kohm, respectively, and using nominal read voltages around 0.3V, the read disturbance is very low [17].

Probabilistic Activation

In order implement the PAF, we leverage the probabilistic-bit (p-bit) design proposed by Camsari et al. [10] and shown in Figure 3c, which is a low-power and compact circuit capable of generating random bits from thermal fluctuations. The p-bit is a Spin-Hall-Effect driven Magnetic Tunnel Junction (MTJ) device with a very low energy barrier [10], which allows thermal agitations to stochastically switch the free layer of the MTJ between its parallel and antiparallel states on sub-nanosecond timescales. Since a current flowing through the bottommost heavy metal layer can effectively bias the free layer of the MTJ, the probability of the MTJ being in HR or LR states can be tuned along a sigmoidal function via the input current, as shown in Figure 3c. By placing a resistor, R_{mid}, having a fixed resistance equal to the average of the HR and LR states of the MTJ between the MTJ and *Vdd*, we can use a voltage divider to switch a pair of inverters, giving us a digital representation of the state of the MTJ, which represents the output of the PAF. Since the energy barrier of the p-bit is very low, it requires current on the order of 100's of nA to bias the PAF, which allows for low-energy BNN calculations using low voltage and highly-resistive scaled RRAM devices for weighted connections.

SIMULATION FRAMEWORK

The simulation framework utilized herein is depicted in Figure 4 and consists of HSPICE modeling of the circuit-level parallel bitwise and bit-counting operations of the RRAM pseudo-crossbar as well as the p-bit PAF for determining the circuit behavior under different RRAM variations and PAF properties, which are then modeled in PyTorch for training and testing on Nvidia Tesla V100 GPU clusters.

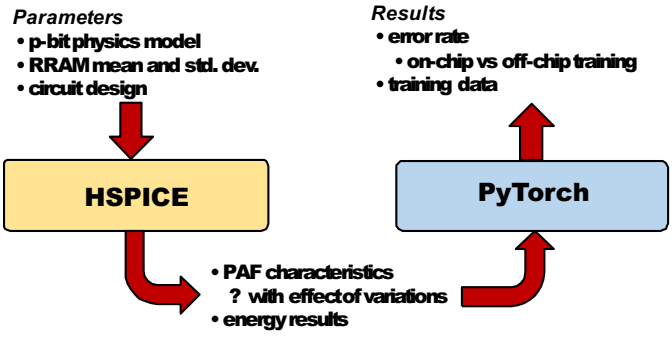

Figure 4: Emerging device model-driven simulation flow.

HSPICE Simulations

On the HSPICE platform utilized, 14nm PTM transistor models [11] were deployed along with RRAM resistance values of $N(5\,M\Omega, \sigma)$ for the LR state, and

$N(50\ M\Omega, \sigma)$ for the HR state, where $N(\mu, \sigma)$ is a normal distribution with a mean of μ and a standard deviation of σ, where σ is varied from 0%-50% of μ in 5% increments, and the p-bit is modeled using experimentally verified physics modules from the Modular Spintronics Library [14] at a *Vdd* of 0.6V. PAF activation probability was measured using Monte Carlo simulations of 100 samples each for 480 different p-bit input currents. The results, shown in Figure 3c, depict the sigmoidal probability of the p-bit's output voltage representing either a '1' or a '0' signaling level.

PyTorch Simulations

For training BNNs using the PyTorch framework, we extend the code developed in [5] to include the impacts of weight-resistance distortion resulting from device variations, as well as implementing our p-bit based PAF. During training, high-precision weight values are stored in RAM, which are used for gradient calculations as per the training algorithm for BNNs developed by Hubara et al [5]. However, the activations in the forward pass are determined strictly by the binary weight values with their associated variations, as if the computation was done in-situ. Although some mismatch may still occur between the high-precision weights used for the backward pass and the in-situ weight variations used for the forward pass, we found that performance is still improved by using the real in-situ weights during the forward pass. The p-bit PAF circuit from the HSPICE simulation is approximated with a probabilistic hard sigmoid function.

The CNN used herein is the same one used for [6], which has six convolutional layers and three fully-connected layers and was trained on the CIFAR-10 dataset [15]. We do not include any pooling layers, however we utilize a stride of 2 on CNN layers 2, 4, and 6. Although it is not explicitly claimed within other BNN hardware accelerator literature, we found that binarizing the final layer results in a significant increase in the error rate, whereby if only the last layer is chosen to be LogSoftmax, the error rate reduces to very tractable levels. Thus, we use LogSoftmax for the final layer, which would be computed using the CPU or ASIC in our scheme. Since the final layer is a small fraction of the total size of the network, the large accuracy improvement can be worthwhile given the acceptable minimal performance degradation incurred.

In-situ vs. Off-Platform Training

When developing BNN accelerator hardware, one has the choice to either train the network Off-Platform using ideal {-1, +1} weight values and then download the final weight configuration to the as-built hardware, or train the network in-situ, using the actual circuit and its associated process variations within the training loop. The latter approach is a distinguishing feature of our design scheme developed herein. We leverage the useful property that device-to-device resistance variations are effectively variations to the {-1, +1} weight values computed when training and testing the network in PyTorch. We compare the performance of in-situ and Off-Platform training techniques with respect to the final error rates under multiple levels of resistance variation, and we show that even with the anticipated rates of variation measured experimentally for fabricated devices, in-situ training allows the learning mechanism to produce BNN configurations that achieve error rates that are nearly identical to the error rates corresponding to idealized networks without device variation.

RESULTS

The results for the simulations conducted as described in the previous section can be summarized in Figure 5 as follows. We simulated at least 100 epochs for in-situ and Off-Platform training with weight variations from 0% to 50% in 5% increments for as-built process variation and/or degradation over the devices' lifetime of operation. The lowest error rate for each condition is shown in Figure 5b, and we show the error rate per epoch for a selection of test cases in Figure 5a. The lowest error rate for ideal weights with no variation is determined to be 15.38%. The increase in error rate due to weight variations for the Off-Platform training condition is negligible under 15% variation, staying within a 2% error rate increase for up to 30% variation, but increases significantly thereafter, reaching a maximum of 12.17% increase in error rate at 50% variation. However, for the in-situ training approach developed herein, the increase in error rate fluctuates within 1% for all weight variations, demonstrating the robustness that is achieved by utilizing the intrinsic device variations within the forward pass for training. Such a result is crucial for deeply-scaled beyond-CMOS devices that exhibit significant process variation. In addition to the error rate analysis, our HSPICE simulations demonstrated that each bit cell that has an input value of '1' consumes an average power of 75 nW, and that the p-bit PAF uses just 4.98 μW, demonstrating a very low-power scheme for accelerating BNNs. The clock speed of the accelerator is dependent upon the retention time of the p-bit, which can be sub-nanosecond [10]. Therefore, with a clock speed of 1 GHz, this implementation uses just 75 aJ per active synapse and 4.98 fJ per PAF calculation. The throughput of this approach, for the training phase, would be limited by the datapaths between CPU/ASIC, RAM, and the accelerator, and as such, it would be critically-dependent upon the implemented components. For the inference phase after the

training phase, the accelerator throughput could be comparable to similar binarized RRAM-crossbar neural network approaches, which is significantly higher than CPU, GPU, FPGA, or CMOS-ASIC approaches [8].

CONCLUSION

The BNN hardware accelerator introduced herein proposed the use of a novel spintronic device for a compact implementation of a PAF that naturally integrates with current-summation-based crossbar and pseudo-crossbar arrays for low-power parallel BNN computations of just 75 nW per activated bit cell and 4.98 uW per PAF. Additionally, we demonstrated that this approach is highly resilient, even to extreme process variation, when utilizing an in-situ training framework. Taken altogether, such a scheme is well-situated for highly-scaled BNN acceleration on resource-constrained platforms such as mobile and IoT.

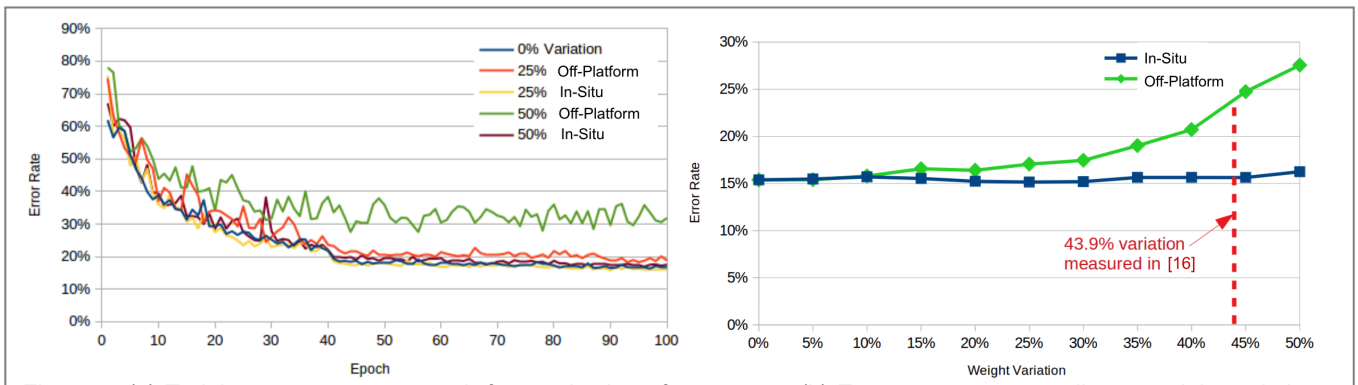

Figure 5: (a) Training error rate per epoch for a selection of test cases; (b) Error rates corresponding to weight variations for both in-situ and Off-Platform training methods.

ACKNOWLEDGEMENTS

This work was supported by the Center for Probabilistic Spin Logic for Low-Energy Boolean and Non-Boolean Computing (CAPSL), one of the Nanoelectronic Computing Research (nCORE) Centers as task 2759.006, a Semiconductor Research Corporation (SRC) program sponsored by the NSF through CCF-1739635. We would like to acknowledge and thank the Advanced Research Computing Center at the University of Central Florida for provision of computing resources used herein.

REFERENCES

1. A. Lecoutre, B. Negrevergne, and F. Yger, “Recognizing Art Style Automatically in painting with deep learning,” *Proceedings of the Ninth Asian Conference on Machine Learning*, PMLR 77, 2017, pp. 327-342.

2. Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” Nature, 521(7553), 2015, pp. 436-444.

3. R. Zand, K. Y. Camsari, S. D. Pyle, I. Ahmed, C. H. Kim, and R. F. DeMara, “Low-Energy Deep Belief Networks using Intrinsic Sigmoidal Spintronic-based Probabilistic Neurons,” *Proceedings of 27th IEEE/ACM Great Lakes Symposium on VLSI* (GLSVLSI-2018), 2018, pp. 15-20.

4. J. Tang, D. Sun, S. Liu, and J. L. Gaudiot, “Enabling deep learning on IoT devices,” *Computer*, 50(10), 2017, pp. 92-96.

5. H. Itay, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. "Binarized neural networks," *Advances in neural information processing systems*, 2016, pp. 4107-4115.

6. X. Sun, S. Yin, X. Peng, R. Liu, J-S. Seo, and S. Yu. "XNOR-RRAM: A scalable and parallel resistive synaptic architecture for binary neural networks." *Proceedings of the 2018 Design, Automation and Test in Europe Conference and Exhibition* (DATE 2018), Vol. 2018-January, 2018, pp. 1423-1428.

7. X. Sun, X. Peng, P-Y. Chen, R. Liu, J-S. Seo, and S. Yu, “Fully parallel RRAM synaptic array for implementing binary neural network with (+ 1,− 1) weights and (+ 1, 0) neurons,” *Proceedings of the 23rd Asia and South Pacific Design Automation Conference*, 2018, pp. 574-579.

8. L. Ni, Z. Liu, H. Yu, and R. V. Joshi, “An energy-efficient digital ReRAM-crossbar-based CNN with bitwise parallelism," *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, Vol. 3, 2017, pp. 37-46.

9. S. Angizi and D. Fan, "IMC: energy-efficient in-memory convolver for accelerating binarized deep neural network." In *Proceedings of the Neuromorphic Computing Symposium* (NCS ‘17), 2017, Article no. 3.

10. K. Y. Camsari, R. Faria, B. M. Sutton, and S. Datta, “Stochastic p-bits for invertible logic,” *Physical Review X,* Vol. 7, Issue 3, 2017, p. 031014.

11. S. Gu, S. Levine, I. Sutskever, and A. Mnih. "Muprop: Unbiased backpropagation for stochastic neural networks." *arXiv preprint* arXiv:1511.05176 (2015).

12. Liu, Beiye, Hai Li, Yiran Chen, Xin Li, Qing Wu, and Tingwen Huang. "Vortex: variation-aware training for memristor x-bar." In *Proceedings of the 52nd Annual Design Automation Conference*, 2015, pp.15.

13. S. Sinha, G. Yeric, V. Chandra, B. Cline, and Y. Cao, “Exploring sub-20nm FinFET design with predictive technology models,” *Design Automation Conference (DAC), 49th ACM/EDAC/IEEE*, 2012, pp. 283-288.

14. K. Y. Camsari, S. Ganguly, and S. Datta, “Modular Approach to Spintronics,” Scientific Reports, Vol. 5, p. 10571.

15. A. Krizhevsky and G. Hinton. *Learning multiple layers of features from tiny images*, tech. report, Univ. of Toronto, Vol. 1, No. 4, 2009.

16. X. Huang, H. Wu, D. C. Sekar, S. N. Nguyen, K. Wang, and H. Qian, “Optimization of TiN / TaO x / HfO 2 / TiN RRAM Arrays for Improved Switching and Data Retention,” Vol. 1C, 2015, pp. 6–9.

17. Jameson, J & Blanchard, P & Cheng, C & Dinh, J & Gallo, A & Gopalakrishnan, V & Gopalan, C & Guichet, B & Hsu, S & Kamalanathan, D. (2013). Conductive-bridge memory (CBRAM) with Excellent High-Temperature Retention. Tech. Digest IEDM. 738-741.

STEVEN D. PYLE is a Ph.D. Candidate in computer engineering at the University of Central Florida (UCF).

JUSTIN D. SAPP is a NSF Research Experiences for Undergraduates (REU) baccalaureate student in computer engineering at UCF.

RONALD F. DEMARA is a professor of electrical and computer engineering at UCF. Contact him at ronald.demara@ucf.edu or visit http://cal.ucf.edu